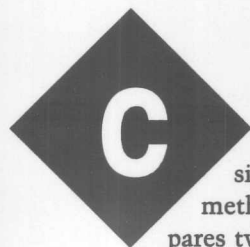


# All About Correlation

## FEATURE ARTICLE

Ron Tipton

If you need to perform signal analysis, Ron takes a graphical approach to explaining the details of correlation. By the time you finish this article, you'll be more than ready to download the software and start working with these data-analysis tools.



Correlation is a signal analysis method that compares two time sequences to find out how alike they are. If the two signals are identical, they are said to have a correlation coefficient ( $R$ ) of unity. If they are completely different, they don't "correlate" at all, and  $R$  is equal to 0. This coefficient ( $R$ ) is a dimensionless number, whose magnitude varies between zero and one.

So, how do you find this coefficient, and what practical use does it have? The method is probably best understood by looking at it graphically.

Suppose you have two sine waves with the same amplitude and frequency drawn on separate pieces of transparent film, placed one on top of the other. If you hold the bottom film stationary and slide the top film to the right or left, you have a simple correlator (see Figure 1).

When the two signal traces coincide,  $R = 1$ . When they are  $90^\circ$  out of phase,  $R = 0$ . (Mathematically, you have a coefficient of  $-1$  when they are  $180^\circ$  out of phase, but for practical reasons, you'll only look at the 0 to 1 interval.)

Although it's not obvious yet, this is a powerful analysis method. You

can construct numeric band-pass filters with narrow passbands for recovering periodic signals buried in noise. Applications include audio reproduction, acoustics, sonar, seismic event detection, and many others.

The examples in this article can be run on a personal computer, and you will be able to use your PC to model virtually any real application by scaling the "real world" signal frequency. Then you can program a digital signal processor if you are building some special-purpose hardware or need the processing speed.

Yes, correlation is a mathematical procedure, so you will have to take a look at one equation to get started:

$$R(m) = \frac{1}{N} \sum_{n=0}^{N-1} x(n)y(n+m)$$

Fortunately, you can relate this equation to the previous graphical example.  $x(n)$  is the sequence of amplitude values in the signal on the bottom film,  $y(n)$  is on the top film, and  $m$  is the variable that describes moving the top film in steps of the signal sampling interval,  $n$ . Each correlation number ( $R$ ) is just the normalized sum of the  $x$  signal multiplied by the  $y$  signal.

### AUTO- AND CROSS-CORRELATION

If you use the same signal for both  $x$  and  $y$ , you have auto-correlation (i.e., a signal correlated with itself).

Cross-correlation is using different  $x$  and  $y$  inputs, and this can provide up to a 20-dB improvement over auto-correlation in pulling a signal out of the noise. This improvement isn't free, so you need to know approximately what the signal looks like without the noise. Let's look at a few examples of

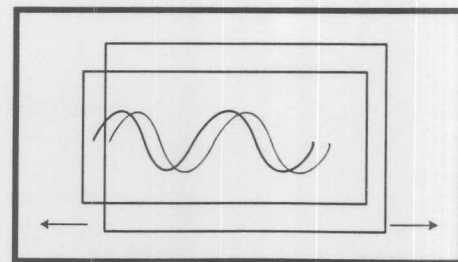
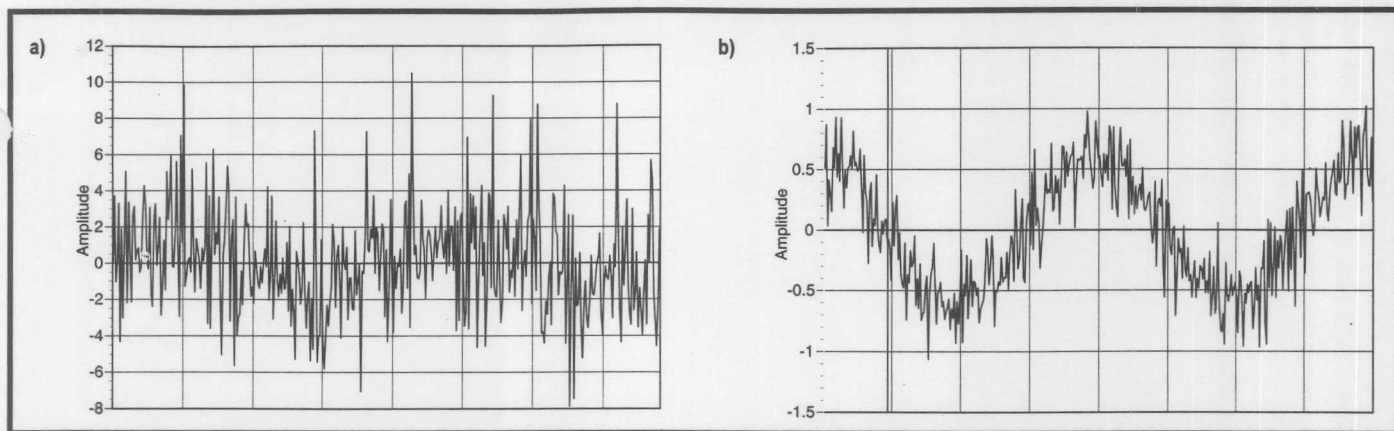


Figure 1—You can make a graphical correlator by plotting wave forms on two sheets of transparent film. Sliding the top film right or left is equivalent to solving the correlation equation.



**Figure 2a**—Here are two cycles of a sine wave with 99% RMS added noise. Auto-correlation (**b**) does not require any knowledge of the signal, but recovery is poorer than is possible with cross-correlation.

auto- and cross-correlation.

Figure 2a shows two cycles of a sine wave with added random noise. The RMS value (see the Root Mean Square sidebar) of the noise is equal to the RMS value of the sine wave, so visually it looks like noise. If you don't know the wave shape and frequency, the best you can do (at least initially) is auto-correlate. Figure 2b shows the results of auto-correlation. You start to see the signal, but it's still noisy. However, if you know the frequency, you can cross-correlate the noisy signal with a noise-free sine wave and get the improvement shown in Figure 3a.

These correlations were done with program `correlat.exe`, a DOS program written in C. The well-commented source code, along with the executable, can be found in `correlat.zip` on the *Circuit Cellar* web site. A companion program,

`gensine.exe` with the C source code, is included to generate input files with and without added noise. The programs are written in MIX Software's PowerC, but they should be easily portable to other compilers if you want to make changes.

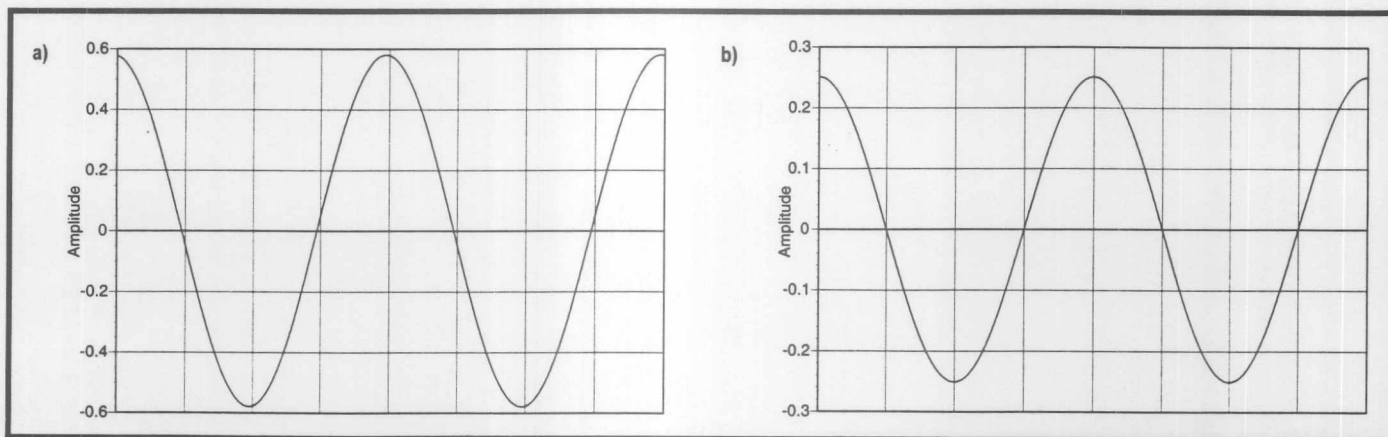
So, what does `correlat.exe` actually do? After asking for input from you, it finds the maximum correlation coefficient and displays it on the screen, along with the corresponding delay between the two inputs. Then, using this delay time, it correlates the input signals and writes an ASCII output file with each value terminated by a new line character. The output file is named either `AUTO.DAT` or `CROSS.DAT` depending on which operation was chosen.

`Gensine.exe` also writes the same kind of output file, which makes these files easy to import into a math program (such as Mathcad) for graph-

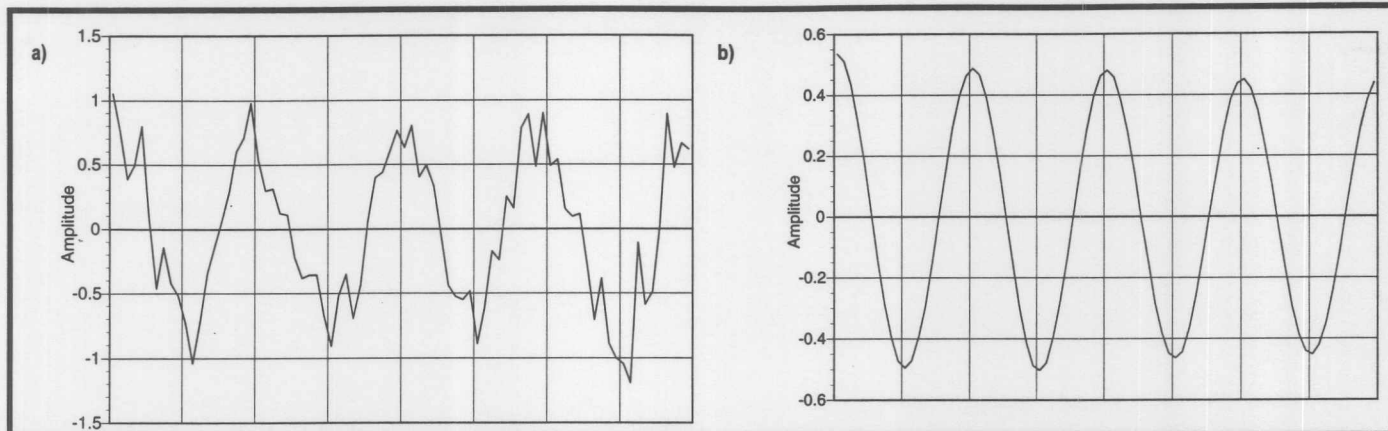
ing the data in reports and other presentations. The output file is named `SINE.DAT`, and it will be written again every time you run the program. So, if you create files containing varying amounts of noise, rename them something descriptive between runs. I wrote all of these programs as development tools, so they lack some of the refinements you expect in commercial software.

To show how powerful cross-correlation can be, I generated an input file setting the RMS value of the noise to be 10 times the RMS value of the sine wave, which is 1000% noise. Auto-correlation shows not a hint of the buried signal, but cross-correlation pulls it out neatly, as you can see in Figure 3b.

Cross-correlation also can be looked at as a type of digital or numeric band-pass filter, and this accounts for its remarkable performance



**Figure 3**—This is the output of cross-correlating a noisy sine wave with a "clean" reference. In **a**), the input had 99% added noise, and in **b**), there was 1000% noise added. Cross-correlation is a powerful tool for pulling signals out of the noise if you can construct a reference signal.



**Figure 4**—Notice the effect of using a slow sampling rate. This (a) is what auto-correlation of a signal with only 50% added noise sampled at 20 points per cycle looks like. The cross-correlation output (b) still shows some distortion from the low sampling rate.

in pulling a signal out of noise. This article isn't meant to be about digital filters, but I included a brief explanation in a second sidebar for those of you who are interested (see the Digital Filters and Correlation sidebar).

In the previous example, I used a sample rate of 200 points per sine wave cycle because it shows dramatic results! A good rule of thumb: Use the highest sample rate you can, within practical limits. As you'll see in a later example, with today's analog to digital converters (ADCs) and DSPs, the highest practical rate can be pretty high.

So, what happens if you have to use a slower sample rate? Figure 4a shows an auto-correlation output of a sine wave with 50% RMS added noise sampled at only 20 samples per cycle. Still noisy, but you can start seeing the periodic signal. By looking at zero crossings, you can estimate the signal's frequency, which let's you use cross-correlation to clean it up as best you can (see Figure 4b). When the signal frequency is unknown or guessed at, simply vary the noise-free signal frequency to minimize the signal distortion out of the correlator.

## PHASE ANGLE RECOVERY

Sometimes the only information needed is the phase angle between two signals at the same frequency, where one signal is corrupted by noise. Cross-correlation excels here. Because the phase measurement resolution depends on the sample rate,

you may need some fast ADCs.

For example, I had to update a data-collection system, which required me to design a cross-correlation phasemeter. The original phasemeter was a counter—one input signal started a high frequency counter, and the second input signal stopped it. The accumulated count was a measurement of the phase difference between the two input signals.

This works well when both signals are "clean," but typically, one of the inputs varies between noisy and extremely noisy. Because it's a constant frequency system, performance is optimized by passing the noisy signal through a narrow band-pass filter and then averaging a number of successive phasemeter counts.

Typically, 8 to 48 measurements are averaged and this improves the phase difference estimate by the square root of the number of averaged counts ( $N$ ). Meaning, the standard

deviation of the jitter is reduced by the square root of  $N$ , but the dynamics of this system prevent increasing  $N$  to more than 48.

Counter phasemeters also have a problem that occurs when the phase difference is near  $0^\circ$  or  $360^\circ$ . When the input signal is noisy, successive counts may jump erratically between zero and full scale. Solving this requires additional logic in the design. However, the correlation meter does not have this problem.

To get an idea of how much improvement I could expect from the correlation phasemeter, I wrote another C language simulation program, `simphase.c`. Along with other parameters, the program asks for the signal amplitude and the amount of noise to add (in RMS percent), which lets us set the input signal to noise ratio (SNR). The second (clean) input is read from a data file generated by the companion program, `genref.c`.

## Root Mean Square

The root mean square, or RMS voltage, is a measure of the energy in a signal. Meaning, one RMS ampere flowing through a resistance produces the same amount of heat as does one DC ampere. For a sine wave, the RMS voltage is equal to 0.707 times the peak voltage.

The RMS value is literally the square root of the sum of the squares of the time-sampled values. Calculating it is an easy way to find the RMS value, if you are looking at the output of an analog to digital converter (ADC). Just square each value and accumulate a running sum for at least as long as the period of the lowest frequency of interest, and then take the square root.

True RMS voltmeters usually use a nonlinear circuit to approximate the relationship between average and RMS. And a few (such as the Hewlett-Packard model 3403C) use a thermal AC to DC converter.



(These programs are included in correlat.zip, which is available for download via the *Circuit Cellar* web site.)

After playing with the simulator for a while, I was convinced that the correlation phasemeter would yield a significant improvement in performance, so I designed and built a prototype. (The clock rate is too high for a successful breadboard, so I laid out a printed circuit board for the initial model.)

The incoming noisy signal is digitized at an 8.5-MHz rate, and the resulting 12-bit words are stored in RAM. (This is 1700 samples per input cycle for a phase resolution of  $0.212^\circ$ .) RAM-1 and RAM-2 are used in a Ping-Pong technique, with RAM-2 being filled while RAM-1 is processed (see Figure 5).

The read and write enable signals for both RAM banks are generated by a timing circuit, which relieves the DSP from having to keep track of

### Digital Filters and Correlation

If the time domain waveform shape is known, an optimal data recovery filter is the convolution between the noisy data and the data's "non-noisy" shape. This means, the filter's coefficients are the noise-free, time-sampled data values in time reverse order.

Time reversal is the only difference between cross-correlation and convolution. If the signal is a sine or cosine wave, then there isn't any practical difference between the forward and reversed time values, so correlation and convolution are the same.

Convolution is sometimes called a "matched filter." A Finite Impulse Response (FIR) digital filter is one way to perform the convolution.

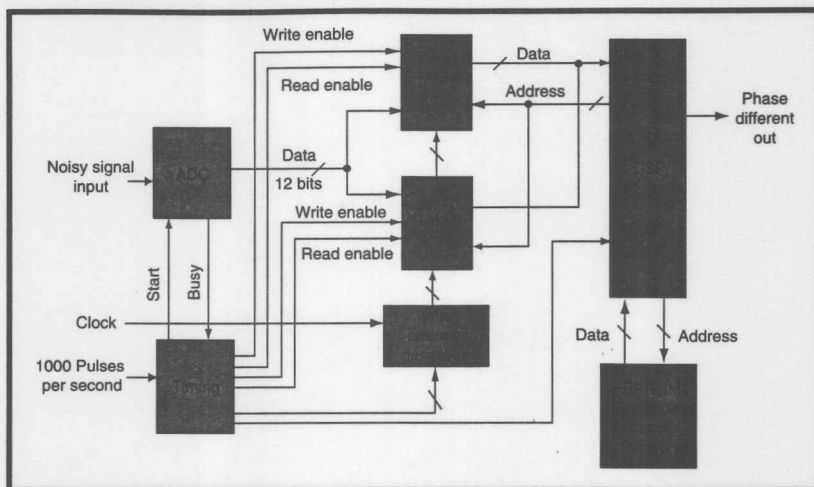


Figure 5—The clean input (reference signal) is read from the EPROM. This meter generates a phase difference output every millisecond using a Motorola fixed-point DSP with a 50-MHz clock.

which RAM bank it is processing. The second (reference) signal values are read from EPROM. Program execution is started by an interrupt signal every millisecond from the timing logic.

The DSP firmware is a bit complex and is divided into two parts. The first part reads the data from RAM-1 or RAM-2 and estimates the count "z" to the next negative-to-positive zero crossing. This count is used as a starting point for the second part of the program, the cross-correlation.

A correlation is run, z is incremented, and another correlation is run. This process is continued until a peak in the correlation coefficient is found. The phase difference or answer is just the delay time corresponding to this maximum. The amount of time saved by starting with an estimated z is crucial to having this phasemeter

run fast enough.

When the input signal is badly corrupted by noise, the zero-crossing estimate may be in error by several counts. Therefore, the second correlation estimate can be smaller than the first one, which means you've missed finding the peak. When this happens, the original z is decremented and used as a new starting point until a peak is found. A

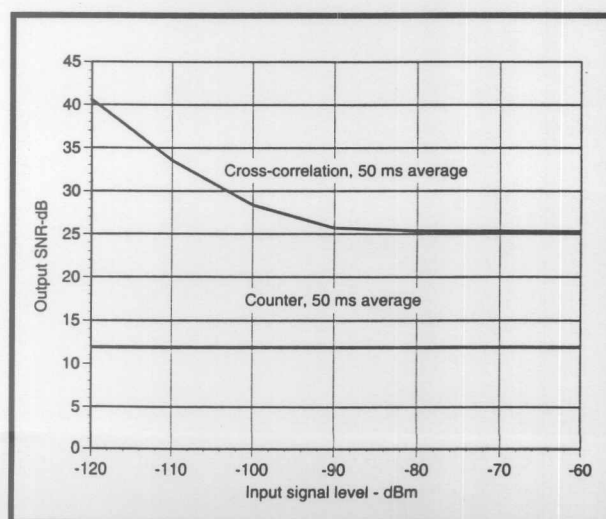
peak is always found in no more than seven correlations, even at minimum SNR. Figure 6 shows the output SNR for both a counter and cross-correlation phasemeter.

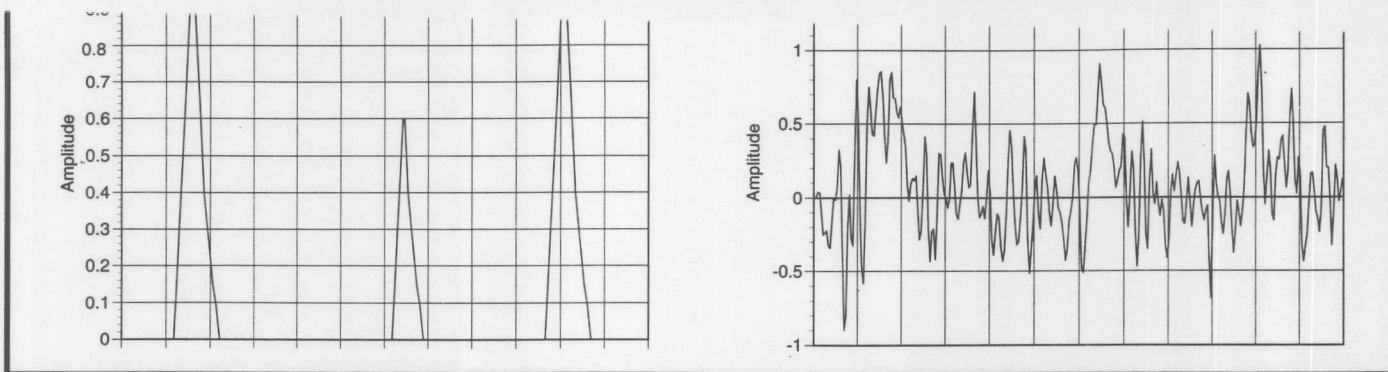
### NON-PERIODIC SIGNALS

Correlation works well in recovering sine waves from noise, because the energy is concentrated at a single frequency, and the noise is broadband (at least when compared to the bandwidth of the signal). A noisy square wave is not recovered well because its signal bandwidth is too broad. The same is true for noisy pulse trains.

Remember that correlation can be thought of as a narrow band-pass filter, so the signal must have most of its energy within this passband. However, a matched filter will do the job.

Figure 6—The cross-correlation phasemeter provides nearly 30-dB SNR improvement for low input signals and is always better than a counter phasemeter. The higher cost of the cross-correlator is justified when performance is a critical requirement.





**Figure 7**—Here you can see an example pulse train (a) and these pulses buried in noise (b). The 30% peak noise was low-pass filtered at 20 kHz by program *genpulse.exe* before the addition.

Figure 7 shows a test group of pulses that I buried in the noise using program *genpulse.exe*. This program assumes a sample rate of 100 kHz, so the noise is low-pass filtered in the program through a 2-pole Butterworth with a 20-kHz cutoff frequency. These filter coefficients are coded into the program, but they can be changed for a different filter if needed. (This filter was designed with the Momentum Data Systems software listed in Resources.)

The matched filter coefficients for recovering the pulses are in file *filter.coe*. They are the sampled amplitude values of one of the pulses in time reversed order and normalized so they sum to unity. Meaning, each coefficient has been divided by its original sum. This is also a plain ASCII file that can be created or changed with any ASCII file editor.

The noisy data file and coefficient file is read by *fir\_bp.exe*, which performs the filtering. The result is shown in Figure 8, and the recovery is

good. You can try adding different amounts of noise and then look at the results. If you look at the source code, you will see that the "process" routines look the same in *correlat.c* and *fir\_bp.c*. Mathematically, correlation and convolution are similar, even though they have different uses.

I suggest that you play with the programs. It's a good way to get a practical appreciation and feel for the data-analysis tools. As I mentioned earlier, they lack commercial polish, but work well and run fast on a Pentium II-class computer at clock speeds of 200 MHz or more. You can also take routines from them to use in writing your own solutions. ☐

*Ron Tipton is an engineer with more than 40 years experience in analog, digital, and software design. He is the president of TDL Technology, Inc., a company he started that specializes in consulting and prototype development. Reprints of his other magazine*

*articles are on the TDL web site. You may reach him at [rtipton@zianet.com](mailto:rtipton@zianet.com).*

## SOFTWARE

All the software mentioned is in *correlat.zip*, which is available on the *Circuit Cellar* web site and also on the TDL site at [www.zianet.com/tld](http://www.zianet.com/tld).

## RESOURCES

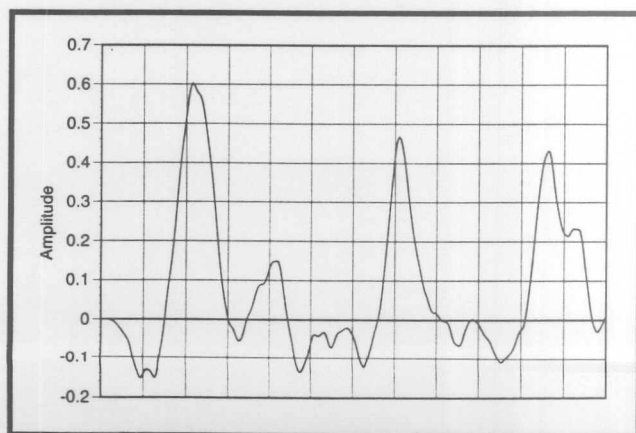
- R.H. Higgins, *Digital Signal Processing in VLSI*, Prentice Hall, Englewood Cliffs, NJ, 1990.
- K.G. Beauchamp, *Signal Processing Using Analog and Digital Techniques*, John Wiley and Sons, 1973.
- C.B. Rorabaugh, *Digital Filter Designer's Handbook: Featuring C Routines*, McGraw-Hill, NY, 1993.

## SOURCES

**PowerC Compiler**  
 Mix Software  
 (800) 333-0330  
 Fax: (972) 783-1404  
[www.mixsoftware.com](http://www.mixsoftware.com)

**Mathcad**  
 MathSoft, Inc.  
 (617) 577-1017  
 Fax: (617) 577-8829  
[www.mathsoft.com](http://www.mathsoft.com)

**Digital filter software**  
 Momentum Data Systems  
 (714) 378-5805  
 Fax: (714) 378-5985  
[www.mds.com](http://www.mds.com)



**Figure 8**—This is what the output of a convolution filter with the noisy pulse train of Figure 7b as the input looks like. Note the pulse time delay as compared to Figure 7a. This is real or "causal" filter even though it's done in software rather than a physical circuit.